

REMARKS

Applicants thank the Examiner for the careful review of the present application and amend independent claims 1, 10, and 16. The amended claims introduce no new matter and are fully supported by the specification. Accordingly, Applicants respectfully request examination of pending claims 1-20.

Claim Rejections Under 35 U.S.C. § 102(e)

The Examiner rejected claims 1-3, 7-13, and 16-20 under 35 U.S.C. § 102(e) as being anticipated by Geva (U.S. Patent No. 6,539,541). Applicants respectfully traverse.

Applicants amend the independent claims to recite “based on the loop structure being often used” and “based on a predetermined number of repeated instructions” to improve the readability of the claims. The amendments introduce no new matter and are fully supported by the Specification. Specifically, “the interpreter 102 keeps track of the number of times a particular byte-code has been interpreted. When this number reaches a predetermined value, the Java virtual machine compiles the byte code... (page 10, lines 16-18).” Further, the Specification discloses, “often-used byte-codes of the Java program are compiled into native code (page 11, lines 10-11).”

In contrast, while Geva teaches unrolling a loop body, Geva does not teach the compilation of a loop structure during the execution of a computer program based on the number of times an interpreter interprets the same code. Consequently, because Geva *does not teach keeping count of often used loop structures*, Geva cannot anticipate independent claims 1, 10, and 16 and the dependent claims that depend

from the independent claims. Thus, Applicants respectfully request the withdrawal of the 35 U.S.C. § 102(e) rejection.

Claim Rejections Under 35 U.S.C. § 103(a)

The Examiner rejected claims 4-6, 14, and 15 under 35 U.S.C. § 103(a) as being unpatentable over Geva as applied to claim 1, in view of Srivastava (U.S. Patent No. 5,457,799). Applicants respectfully traverse.

The claimed invention builds a loop tree for loop unrolling. The specification discloses, “[t]he loop tree represents the loops of the Java program by branching loops from a root node. The child nodes of a loop node represent nested loops, while loops on the same level as a loop node represent parallel loops (page 14, lines 13-16).” Subsequently, the loop tree is used during the interpretation and compilation of a computer program (independent claims 1 and 10).

Srivastava discloses an optimizer for loops in a computer program by building a graph of nodes and vertices such that all the nodes are procedures in the computer program and the vertices indicate the flow of execution among the procedures (col. 1, lines 60-65). If a procedure repeats itself, then a first node representing the procedure and all the subsequent nodes connected to the first node by vertices is called a loop region (*see* col. 1, lines 66-67 through col. 2, lines 1-4; Figure 2; col. 3, lines 35-57). Thus, the loop region includes multiple nodes such that the first node indicates the beginning of a loop.


Srivastava then discloses “[i]f the program call graph 300 were to have the form of a tree, the loop region of the tree would include the called procedure 321 and all its descendants, for example the procedures 321, 322, 323, 324, 325, and 329. However, *very few real programs have call graphs in the form of a tree* (col. 4, lines 11-16).” Thus, although Srivastava discloses a tree, Srivastava also discloses that the tree is used in *very few real programs*. Thus, the Office’s statement that “using loop regions in the form of trees makes calling more efficient (paper #3, page 4, regarding claim 4)” is inconsistent with the reference because efficient structures/processes are typically used in real programs. Generally, one of ordinary skill in the art would not use an inefficient structure/process in real programs.

Further, as shown by Figure 2 in the reference, programmers use an editor to create computer code, which is then compiled by a compiler 30 to produce object code for the target computer. The object code is then linked and executed by the computer. However, Figure 2 and the corresponding text *do not disclose an interpreter* that keeps a count of instructions such that *instructions that are repeated a predetermined number of times are compiled* into native code.

Specifically, the teachings of Geva cannot be combined with the teachings of Srivastava to teach or suggest the limitations of claims 1, 4-6, 10, 14, and 15 as a whole. Thus, one of ordinary skill in the art would not have been motivated to use a loop tree as recited in claims 4-6, 14, and 15 based on the combined references. Because Geva and Srivastava singly or in combination cannot teach or suggest to one of ordinary skill in the art the limitations recited in claims 1, 4-6, 10, 14, and 15, Applicants respectfully request the withdrawal of the 35 U.S.C. § 103(a) rejection.

Applicants respectfully request a Notice of Allowance based on the foregoing remarks. If the Examiner has any questions concerning the present amendment, the Examiner is kindly requested to contact the undersigned at (408) 749-6900. If any other fees are due in connection with filing this amendment, the Commissioner is also authorized to charge Deposit Account No. 50-0805 (Order No. SUNMP017). A copy of the transmittal is enclosed for this purpose.

Respectfully submitted,
MARTINE & PENILLA, LLP



Feb Cabrasawan
Reg. No. 51,521

Martine & Penilla, LLP
710 Lakeway Drive, Suite 170
Sunnyvale, California 94086
Tel: (408) 749-6900
Customer Number 32291